ANL/ALCF/ESP-17/5

First-Principles Simulations of Functional Materials for Energy Conversion

Technical Report for the ALCF Theta Early Science Program

Argonne Leadership Computing Facility

ALCF Early Science Program (ESP) Technical Report

ESP Technical Reports describe the code development, porting, and optimization done in preparing an ESP project's application code(s) for the next generation ALCF computer system. This report is for a project in the Theta ESP, preparing for the ALCF Theta computer system.

ANL/ALCF/ESP-17/5

First-Principles Simulations of Functional Materials for Energy Conversion

Technical Report for the ALCF Theta Early Science Program

edited by
Timothy J. Williams and Ramesh Balakrishnan

Argonne Leadership Computing Facility

prepared by
Huihuo Zheng, Christopher Knight, Marco Govoni, Giulia Galli, and Francois Gygi

September 2017

This work was supported in part by the Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

First-Principles Simulations of Functional Materials for Energy Conversion

Huihuo Zheng*[1], Christopher Knight†[1], Marco Govoni‡[2, 3], Giulia Galli§[2, 3], and Francois Gygi¶[4]

[1] Leadership Computing Facility, Argonne National Laboratory, Argonne, IL
[2] Institute for Molecular Engineering and Materials Science Division, Argonne National Laboratory, Argonne, IL
[3] Institute for Molecular Engineering, University of Chicago, Chicago, IL
[4] Department of Computer Science, University of California, Davis, Davis, CA

November 19, 2018

1 Introduction

Computational modeling has become a very effective approach in predicting properties of materials, and in designing functional materials with targeted structural, thermal, or optical properties. Many electronic structure methods have been developed, including density functional theory (DFT) (see [1] and the references therein), many DFT-based methods [2, 3], quantum Monte Carlo [4], and quantum chemistry methods [5]. Among them, DFT has been widely used in physics and materials science because it is computationally cheaper than other methods but still gives desired accuracy. It also provides a good starting point for higher levels of theory, such as many-body perturbation theory and quantum Monte Carlo.

In parallel to these electronic structure method developments, a dramatic increase in computing capabilities over the last decade has enabled large-scale electronic structure calculations to address leading-edge materials science problems. In particular, with Theta at the Argonne Leadership Computing Facility (ALCF), our early science project investigated large-scale nanostructured materials for energy conversion and storage using two open-source electronic structure codes Qbox (http://qboxcode.org) and WEST (http://west-code.org). Qbox is an ab-initio molecular dynamics code based on plane wave DFT, and WEST is a post-DFT code for excited state calculations within many-body perturbation theory.

Theta is a Cray/Intel system based on 2nd generation Xeon-Phi processors (code-named Knights Landing (KNL)) and serves as a bridge between the current flagship machine, Mira, a 10 petaflop IBM Blue Gene Q, and the upcoming ALCF-3 Aurora flagship machine. Theta has several distinct

*huihuo.zheng@anl.gov
†knightc@anl.gov
‡mgovoni@anl.gov
§gagalli@uchicago.edu
¶fgygi@ucdavis.edu

architectural features, such as Advanced Vector Extensions (AVX-512) and a memory hierarchy, which are expected to enable scientific codes to achieve higher performance. Other key differences between Theta and Mira are the networks (IBM's 5D torus vs. Cray Aries interconnect with 3-level Dragonfly topology) and the overall balance of compute/memory capabilities relative to network performance. It is crucial to understand the combined impact of these technologies on an application's performance (with production quality inputs) in order to scale efficiently to large fractions of the machine and improve the time-to-solution of science-critical calculations. In this report, we summarize our efforts to improve performance of both Qbox and WEST on Theta, some of which have also resulted in improved performance on Mira. We expect the optimizations accomplished during the Theta Early Science project will play a key role in refining the road-map for software development efforts targeting future large-scale computing resources (e.g. Aurora at ALCF).

The contents of this report are organized as follows. In Section 2, we give a brief summary of the science objectives explored in this early science project. In Section 3, we introduce the Qbox and WEST application codes and briefly discuss the underlying theories and algorithms – plane wave DFT and many-body perturbation theory. Then, in Section 4, the optimization efforts completed for both codes are discussed. Finally, in Section 5, we comment on the portability of the two codes, in particular highlighting the possibility for workflows to utilize both Mira and Theta as enabled by this project.

2 Science Summary

In this project, the electronic properties of several nanostructured materials were investigated for their use in solar and thermal energy conversion devices. In particular, we focused on the opto-electronic properties of inorganic nanostructured samples for use in third generation solar cells, i.e. solar cells exploiting intermediate band transitions and/or multi-exciton generation for carrier multiplication. The goal is to use advanced theoretical methods to improve predictions of the properties of new energy materials thereby accelerating discovery of materials for use in future devices and helping to optimize device performance.

We focus on materials exhibiting complex structures on multiple length scales and predict their electronic and thermal properties using a theoretical framework that combines *ab initio* molecular dynamics with accurate electronic structure methods. The molecular simulations are employed to compute ensemble averages of thermodynamic and transport properties of nanostructured materials with complex morphologies and compositions, inclusive of interfaces between nano- and meso-scale building blocks (see Fig. 1). The same atomistic trajectories subsequently serve as input to many-body perturbation theory calculations to compute electronic properties, including band gaps, band edges, absorption spectra, and dielectric properties.

We collaborate with experimental groups (V. Klimov at LANL and D. Talapin at UoC) to determine initial structure models to be used as input for *ab initio* molecular dynamics simulations for further structural optimization, and subsequently many-body perturbation calculations to determine accurately the optoelectronic properties. We focus on group IV and III-V nanoparticles with a variety of ligands, which are currently being tested experimentally. In addition, we compute band gaps and energy level alignment between ligands and constituent nanoparticles (NPs), or between the two constituents in a heterogeneous interface system, so as to optimize both quantities to increase the system hole and electron mobilities. Together, these important microscopic pieces of information

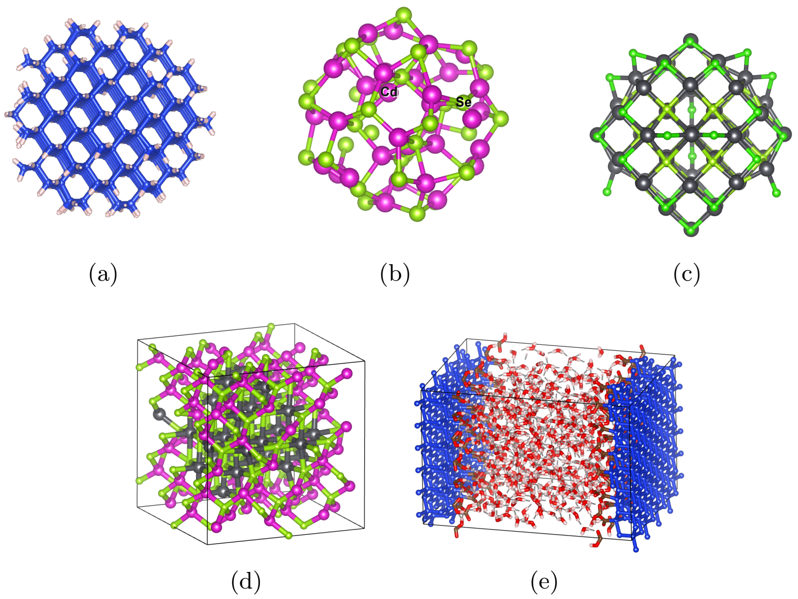


Figure 1: Examples of systems investigated in this project using the Qbox and WEST codes on Theta: (a) Silicon nanoparticle; (b) CdSe nanoparticle; (c) PbSe nanoparticle; (d) Pb nanoparticle imbedded in a CdSe ligand; (e) Silicon/water interface.

combine to provide guidance on the design of materials for high-efficiency solar cells.

3 Codes, Methods, and Algorithms

3.1 Qbox – plane wave density functional theory

Qbox is an *ab initio* molecular dynamics code based on the DFT with plane waves and pseudopotentials formalism. In DFT, the electronic properties of a system are described by the following self-consistent Kohn-Sham equation [1],

$$\hat{h}_{KS}\psi_i = \left[-\frac{1}{2}\nabla^2 + V_{\text{ext}}(\mathbf{r}) + \int d\mathbf{r}' \frac{\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} + V_{xc}[\rho(\mathbf{r})] \right] \psi_i(\mathbf{r}) = \epsilon_i \psi_i(\mathbf{r}), \quad i = 1, \cdots N\,. \quad (1)$$

Here, $\rho(\mathbf{r})$ is the density of electrons in the system, $\psi_i(\mathbf{r})$ is a single particle orbital, and ϵ_i is the Kohn-Sham energy associated with $\psi_i(\mathbf{r})$. The first and second terms are, respectively, the kinetic energy and the potential energy including the Coulomb interaction from ions. The third and forth terms represent electron-electron interactions via the Hartree and exchange correlation V_{xc} potentials, respectively.

The exchange-correlation potential V_{xc} is a functional of the electron density $\rho(\mathbf{r})$, the exact form of which is unknown. Different approximate forms have been introduced in the literature [1]: the local density approximation (LDA) [6], local spin density approximation (LSDA), various forms of generalized gradient approximation (GGA) [7, 8, 9], meta-GGA [10], and many flavors of hybrid functionals, such as PBE0 [11] and HSE [12] that include exact Hartree-Fock exchange. Unfortunately, there is no known systematic way to achieve an arbitrarily high level of accuracy and different functionals might be suitable to specific classes of systems. It has been shown that hybrid functionals are good for systems involving transition metal elements, however, these functionals come with a dramatic increase in computational expense as we will see later on.

The Kohn-Sham equations are usually solved iteratively. One first constructs the Kohn-Sham Hamiltonian with an initial guess of wave functions and density, solves the Kohn-Sham eigenvalue problem obtaining new wave functions and density, inserts the new wave functions and density into the Kohn-Sham Hamiltonian, and solves the equations again. This process is repeated until self-consistency is achieved.

The orbitals, $\psi_i(\mathbf{r})$, are usually expressed in a finite basis set (e.g. plane waves), such that the differential form of Kohn-Sham equations could be rewritten into a linear algebraic form. In the plane wave basis, the system is modeled in a simulation box (e.g. unit cell of crystal or sufficiently large volume enclosing an isolated cluster) with periodic boundary conditions. At the Γ-point, the wave function is represented as

$$\psi_i(\mathbf{r}) = \frac{1}{N} \sum_{\mathbf{G}} e^{i\mathbf{G}\cdot\mathbf{r}} \psi_i(\mathbf{G}) \, . \tag{2}$$

The transformation between real space $\psi_i(\mathbf{r})$ and reciprocal space $\psi_i(\mathbf{G})$ is conveniently done through 3D fast Fourier transformations (FFT). Efficient on-the-fly transformations are necessary because some terms in Eq. (1) are evaluated more efficiently in reciprocal space (e.g. kinetic operator) or real space (e.g. V_{xc}).

The implementation of the plane wave DFT algorithm in Qbox is based on the hybrid MPI+OpenMP parallelization model [13, 14, 15]. Processors are logically arranged as a 2D grid with occupied orbitals (bands) distributed across processors as shown in Fig. 2. The plane wave coefficients for each band are distributed across processors within a column process group. This decomposition has many advantages (e.g. column- and row-wise communication patterns) [13, 14, 15]. In particular, 3D FFTs only involve communication within columns of the process grid, whereas the accumulation of partial charge density contributions only involves communication within rows of the process grid. This approach avoids, as much as possible, the use of global communication involving all processors. In particular, it allows many 3D FFTs to be performed simultaneously (one for each band) within each column process group. This makes the hybrid-DFT calculation (involving massive 3D FFT computations) highly scalable. However, one can also immediately note intrinsic strong-scaling limits introduced (e.g. the number of columns should not exceed the number of occupied bands). When a 1D decomposition is used for 3D FFTs (slab+pencil), the number of processes in a column should not exceed the grid dimension along the z-axis. Therefore, an intrinsic hard scaling limit of $nproc \sim N_{band} \times N_z$ is to be expected. In implementations, however, it may not be beneficial to run in this limit due to communication overheads from small message sizes relative to the amount of computational work per processor.

Qbox uses BLAS, LAPACK and ScaLAPACK for dense linear algebra. For 3D FFTs, Qbox uses its own parallel data layout and calls 1D and 2D transform functions via an FFTW interface (supported by FFTW, MKL, and ESSL, for example) or IBM's ESSL library (available on IBM BG/Q, for example). Both parallel dense linear algebra and 3D FFTs have communication overheads which might potentially affect the strong scaling performance of applications, such as Qbox. Our optimization work in this project largely focused on reducing or hiding these communication overheads, thus improving performance in the strong-scaling limit of Qbox.

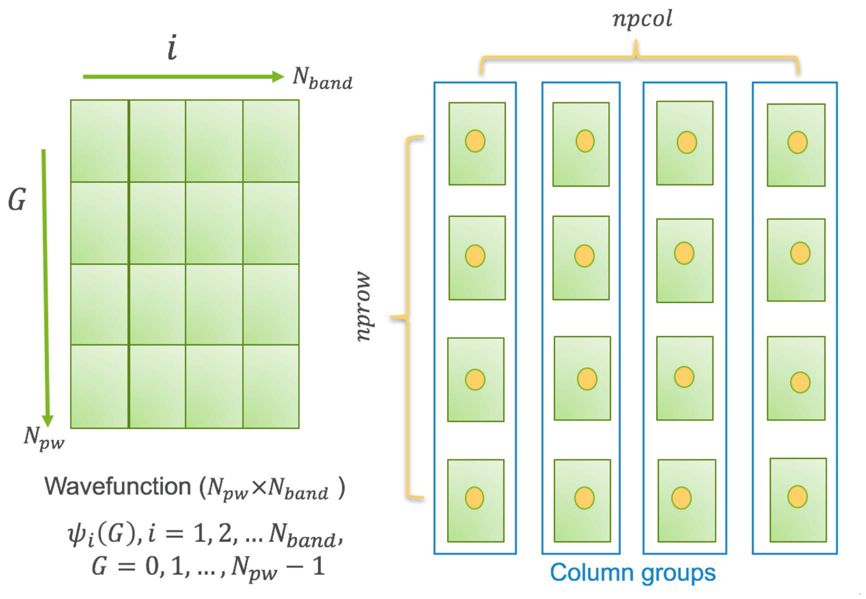

npcol
i
N_{band}
G
N_{pw}
nprow
Wavefunction ($N_{pw} \times N_{band}$)
$\psi_i(G), i = 1, 2, \dots N_{band},$
$G = 0, 1, \dots, N_{pw} - 1$
Column groups

Figure 2: Distribution of wave functions on a 2D processor grid of $nprow \times npcol$. N_{band} is the number of bands, and N_{pw} is the number of plane waves. Different column groups own different groups of wave functions (bands).

3.2 WEST – many-body perturbation theory based on GW approximation

While DFT is good for describing ground state properties of materials, the theory itself does not provide a direct connection to excited states. It has been demonstrated in many materials that the Kohn-Sham gap (using LDA and GGA functionals) is smaller than the experimental optical gap (the energy needed to excite an electron from the top of the valence band to the bottom of the conduction band) [16]. Therefore, to get an accurate description of excited state properties, one needs higher level post-DFT theories. The many-body perturbation theory at the GW level has been proven to be a very good theory for excited states calculations [17, 3]. WEST is a post-DFT code based on many-body perturbation theory with computationally efficient algorithms and implementations to access excited states in materials.

The GW theory is based on the quasiparticle concept and Green's function methods [17]. In a Coulomb system, the repulsion between electrons leads to a depletion of negative charge around a given electron. The ensemble consisting of this electron and its surrounding positive screening charge forms a quasiparticle. The quasiparticle energy is determined by the following equation,

$$\hat{h}\psi(\mathbf{r}) = -\frac{1}{2}\nabla^2\psi(\mathbf{r}) + v_{\text{ext}}(\mathbf{r})\psi(\mathbf{r}) + v_H(\mathbf{r})\psi(\mathbf{r}) + \int \Sigma(\mathbf{r}, \mathbf{r}', \epsilon)\psi(\mathbf{r}') = \epsilon\psi(\mathbf{r}), \tag{3}$$

where v_H is the Hartree potential, Σ is the self-energy and a non-local operator that describes exchange and correlation effects beyond the Hartree-Fock approximation or DFT [2]. ψ is the quasiparticle wave function and ϵ, in general, is a complex quantity with its real part corresponding to the quasiparticle energy and its imaginary part determining the quasiparticle lifetime.

Eq. (3) is very similar to the Kohn-Sham equation [see Eq. (1)] except that the exchange-correlation functional is replaced by the self-energy Σ. Within the GW framework, the self energy is approximated as a product of the Green's function G and screened Coulomb interaction W (hence the name GW).

$$\Sigma(\mathbf{r},\mathbf{r}',\omega) = i\int_{-\infty}^{\infty}\frac{d\omega'}{2\pi}G(\mathbf{r},\mathbf{r}',\omega+\omega')W(\mathbf{r},\mathbf{r}',\omega') \tag{4}$$

W is directly related to the density-density response of the material $\chi(\mathbf{q},\omega)$. We here omit the exact expressions for G and W, which are found in Ref. [18].

The WEST code is organized as a driver for DFT codes and the version 2.0.0 (focus of this work) is currently coupled to the plane wave DFT Quantum Espresso code (version 5.4.0; http://www.quantum-espresso.org/). Key features distinguishing WEST from other GW implementations include the following [18]: (1) the use of projective dielectric eigenpotentials (PDEPs)[19, 20] as basis set to represent the density-density response function which allows to avoid large matrix inversion operations, (2) the use of density functional perturbation theory [21] to compute PDEPs without explicit summations over empty (virtual) states, and (3) the use of the Lanczos algorithm to compute the frequency dependent G and W, thus eliminating the need for plasmon pole models. Together with an efficient parallel implementation, these algorithms enable WEST to do large-scale GW calculations.

There are two major steps in a WEST calculation.

(1) *wstat*: iterative diagonalization of the static (zero frequency) density-density irreducible response function $\chi_0(\mathbf{q},0)$. The resulting eigenvectors will be used as a basis for expanding $\chi(\mathbf{q},\omega)$ and $W(\mathbf{q},\omega)$.

(2) *wfreq*: full frequency GW calculation including computation of $\chi = \chi_0 + \chi_0 v\chi$ (v is the Coulomb potential), and $\Sigma = iGW$, from which the quasiparticle energy spectrum is obtained.

In the following, we only discuss those algorithms in WEST that were the focus of our optimization efforts. Interested readers can refer to Ref. [18] for a more complete description of the formalism and implementations of *wstat* and *wfreq*.

In *wstat*, one computes the response of the system, change of charge density $\Delta\rho_i$, due to external perturbations ΔV_i within the linear response framework (see Fig. 3),

$$\Delta\rho_i = \chi\Delta V_i\,, \tag{5}$$

The Davidson algorithm is used to iteratively diagonalize the response function. Eq. (5) serves as the matrix-vector multiplication rule, where the response function is never represented as a $N_{pw}\times N_{pw}$ matrix. A low-rank decomposition of the response function (of size $N_{pert}\times N_{pert}$) can be achieved using the N_{pert} most screened eigenvectors (with $N_{pert} \ll N_{pw}$). In our implementation, the communication between *nproc* MPI processes is organized into a two dimensional grid: one dimension of the grid distributes perturbations and the other distributes plane-wave coefficients and FFT operations. This scheme exploits the embarrassing parallelism carried by Eq. 5 in the perturbation index, and results in an inherit strong scaling limit of $nproc = N_{pert}\times N_z$, where N_z is the FFT grid size in the z direction[18].

We also note that in Eq. 5 one can express the total change of the density of N_{occ} electrons in terms of the sum of N_{occ} single particle contributions, namely:

$$\Delta\rho_i(\mathbf{r}) = \sum_{\sigma}\sum_{n=1}^{N_{\mathrm{occ}}^{\sigma}}\int_{\mathrm{BZ}}\frac{d\mathbf{k}}{(2\pi)^3}\left[\Psi^{i*}_{n\mathbf{k}\sigma}(\mathbf{r})\Delta\Psi^{i}_{n\mathbf{k}\sigma}(\mathbf{r}) + c.c.\right]. \tag{6}$$

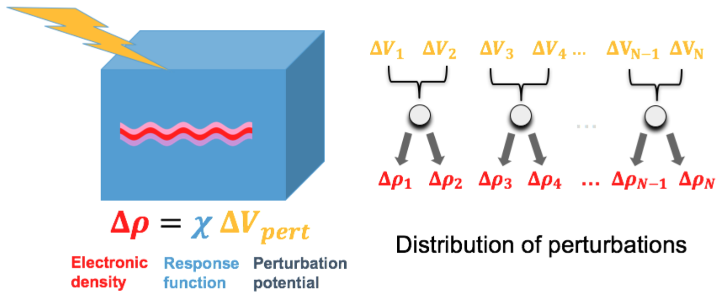

$\Delta\rho = \chi\,\Delta V_{pert}$
Electronic density
Response function
Perturbation potential
ΔV_1 ΔV_2 ΔV_3 ΔV_4 ... ΔV_{N-1} ΔV_N
$\Delta\rho_1$ $\Delta\rho_2$ $\Delta\rho_3$ $\Delta\rho_4$ ... $\Delta\rho_{N-1}$ $\Delta\rho_N$
Distribution of perturbations

Figure 3: Schematic of *wstat* calculations. The response of the system to N_{pert} perturbation is an embarrassingly parallel problem.

The linear change of single-particle Kohn-Sham orbitals ($\Delta\Psi^{i}_{n\mathbf{k}\sigma}$) is obtained by solving the Sternheimer equation [21, 22] for each occupied orbital, identified by band index n, k-point $\mathbf{k}$ and spin polarization σ.

4 Code Development

4.1 Optimization of Qbox

In Qbox, the majority of runtime is spent in math libraries (e.g. ScaLAPACK and FFTW). Therefore, application performance to a large degree depends on external libraries tuned to specific architectures and effectively utilizing resources. Thus, the focus of our optimization efforts have largely been directed at improving performance of the algorithms and addressing communication overheads to improve the overall strong scaling efficiency of Qbox. One area of importance is efficient computation of eigenvalues and eigenvectors, for which the ScaLAPACK eigenvalue solver was replaced with one from ELPA (Eigenvalue SoLvers for Petaflop-Applications) for performance improvements in the range 5-10x depending on problem size and solver [23, 24]. A second area of importance that we investigated was understanding the criteria necessary to setup optimal processor configurations for the simultaneous best performance of 3D FFTs and dense linear algebra [14]. Competing effects were observed for any single data layout, thus we swap data between two layouts on-the-fly to reduce communication overheads in 3D FFTs and dense linear algrebra computation improving substantially the strong scaling parallel efficiency of Qbox.

4.1.1 Replacing ScaLAPACK eigenvalue solver with ELPA

Solving the eigenvalue problem can be a crucial component of DFT depending on the physics of the system being investigated (e.g. low band gaps and metallic systems) and methods employed (e.g. wave function optimizer). For example, when using the Jacobi-Davidson (JD) algorithm to update the wave function coefficients at each iteration, an eigenvalue problem needs to be solved for matrices of size $2N_{band} \times 2N_{band}$. Therefore, choosing an efficient eigenvalue solver is crucial to the overall performance of the code and enabling calculations on increasingly larger system sizes.

Qbox originally used subroutines from ScaLAPACK as the default eigenvalue solvers, *syev* (for real matrices) and *heev* (for complex matrices). Many alternative algorithms have been proposed to replace the ScaLAPACK solvers and recently the ELPA library [23, 24] has been shown to outperform ScaLAPACK at both small and large scales. We therefore implemented an interface to ELPA in Qbox. In the following, we present our benchmark studies of several algorithms in ScaLAPACK and ELPA for problem sizes relevant to large-scale calculations in Qbox.

4.1.1.1 Performance of ELPA in comparison with ScaLAPACK

We focused our study on square symmetric matrices with dimensions ranging from 512×512 to $4,096 \times 4,096$, which are most relevant to the production calculations in our Theta Early Science project. Strong scaling analyses were performed by varying the number of MPI processes from 1 to 16,384 for both Theta and Cetus (a 4–rack IBM BG/Q system). For the sake of brevity, results are only shown for a $2,048 \times 2,048$ matrix (similar behavior is observed for other matrix sizes).

There are several factors that potentially affect the performance of ELPA that need to be considered. Like ScaLAPACK, ELPA does rely on LAPACK and BLAS subroutines for serial portions of algorithms, thus math libraries tuned for specific architectures are required for optimal performance. We studied this dependence on Theta by linking ELPA to Intel MKL and Cray LIBSCI libraries. Another factor to consider is the data layout, in particular, the blocking factor of the matrix. The data layout for ELPA relies on the same block-cyclic distribution across a rectangular processor grid as ScaLAPACK (thus making it straightforward to create an interface to ELPA), but the optimal choice of blocking factor for the matrix needs to be investigated as the optimal blocking factor for the two libraries may be different.

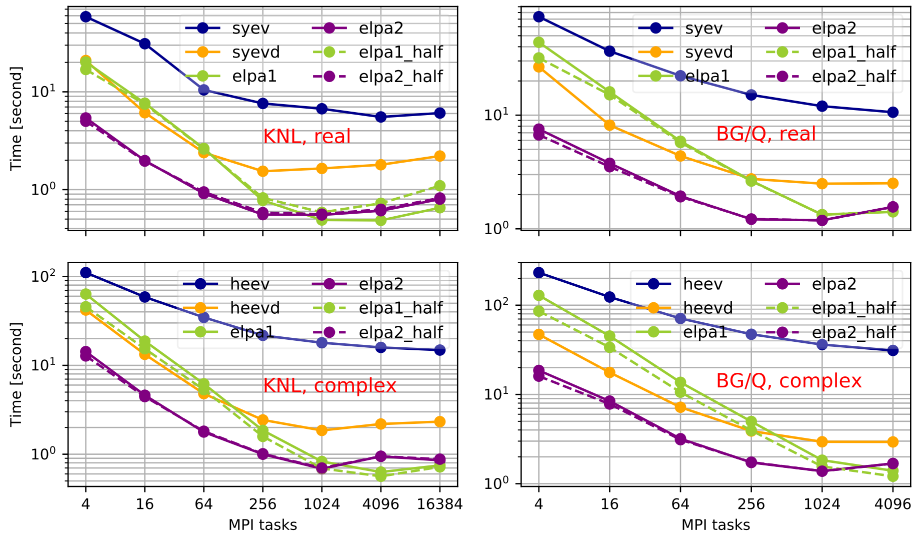


Figure 4: Performance of ELPA compared with ScaLAPACK on Theta (KNL) and Cetus (BG/Q) for $2,048 \times 2,048$ matrices. In all cases, square processor grids were used (i.e. $nprow = npcol = \sqrt{nproc}$), and blocking factors were set as $mb = m/nprow$ and $nb = n/npcol$. On both Theta and Cetus, 1 MPI rank per core was used (64 and 16 per KNL and BG/Q node, respectively). *elpa1* and *elpa2* are ELPA stage 1 and 2 algorithms, respectively, with all eigenvalues and eigenvectors computed. *elpa1_half* and *elpa2_half* are similar, but with only half of the eigenvalues and eigenvectors computed. Cray LIBSCI and IBM ESSL were used on Theta and Cetus, respectively, as the underlying math library.

Figs. 4 show the performance of ELPA on BG/Q and KNL, where on both platforms we observed the following.

(1) The divide and conquer algorithm *syevd* performs better than *syev*, and similarly for *heevd* compared to *heev*.

(2) The ELPA stage 2 algorithm (*elpa2*) performs better than ScaLAPACK solvers and is generally one order of magnitude faster than *syev*/*heev*.

(3) *elpa2* performs better than *elpa1* on smaller numbers of processes, whereas both perform similarly as number of processes increases. This is due to *elpa1* being sensitive to the choice of the blocking factor. With an appropriately tuned blocking factor, *elpa1* was observed to perform better than *elpa2* even at small processor counts, as discussed below.

(4) *elpa2* was observed to have strong scaling properties similar to ScaLAPACK's *syev*, *syevd*, *heev*, and *heevd*. On both BG/Q and KNL resources, these algorithms scale well up to about $(m/64) \times (m/64)$ processor grids for the matrix sizes examined. As the number of processors is increased, the size of the local matrix block owned by a process decreases in size. The scaling benchmarks on both Theta and Cetus indicate that the eigenvalue solvers scale well down to 64×64 local matrices before communication costs degrade performance.

(5) ELPA performs better on KNL compared to BG/Q if identical number of processors are used. Factors contributing to improved performance on KNL include higher achievable flops (e.g. dgemm) and larger memory bandwidth via MCDRAM.

(6) Fig. 5 shows that ELPA on Theta performs better when linked against Intel MKL compared to Cray LIBSCI, although the differences between ELPA algorithms (*elpa1* and *elpa2*) is much smaller than that of ScaLAPACK algorithms (*syev* and *heev*). In both cases, the ELPA algorithms are observed to outperform ScaLAPACK in most cases regardless of whether Intel MKL or Cray LIBSCI is used.

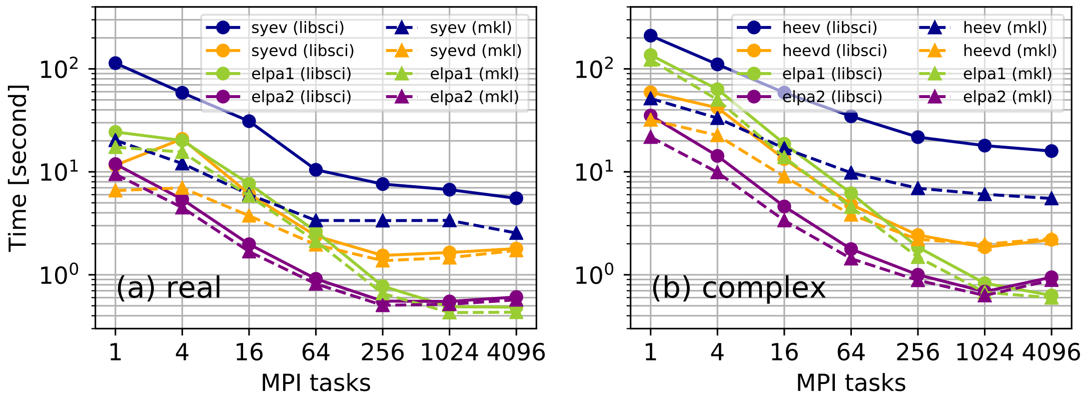


Figure 5: Performance of ELPA on Theta with different math libraries (Intel MKL vs Cray LIBSCI). The size of the matrix is $2,048 \times 2,048$. In all the cases, $nprow = npcol = \sqrt{nproc}$, and blocking factors are set as $mb = m/nprow$ and $nb = n/npcol$.

(7) In previous studies, blocking factors were always defined as $mb = m/nprow$, $nb = n/npcol$, but in general one can choose to use smaller blocking factors. Using a 2048×2048 matrix,

we investigated the effect of blocking factor on performance with two process configurations ($nprow = npcol = 8$ and $nprow = npcol = 32$) and varying the blocking factor from 1 to $n/npcol$.

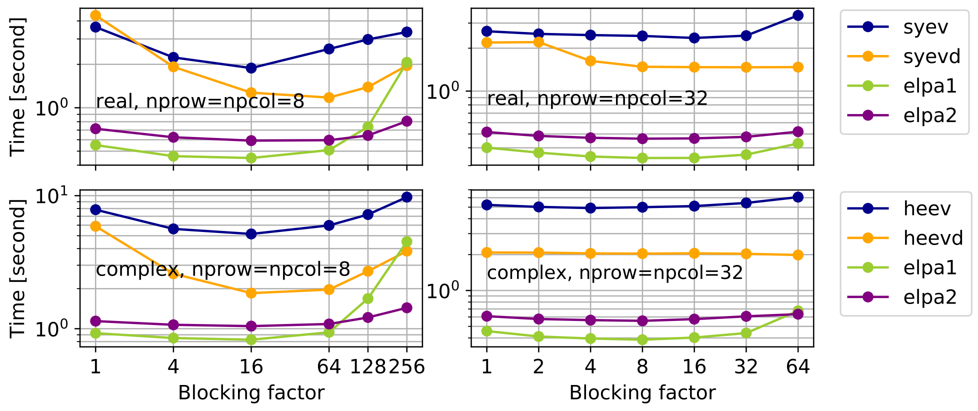


Figure 6: Performance of ELPA compared with ScaLAPACK on Theta for different blocking factors. The size of the matrix is $2,048 \times 2,048$. The blocking factors are varied from 1 to $nb = n/npcol$. MKL is used as the underlying library.

As shown in Fig. 6, we observed that *elpa1* is more sensitive to choice of blocking factor compared to *elpa2*. The performance of *elpa1* degrades severely if $nb > 64$, whereas it performs slightly better than *elpa2* for $nb < 64$. This is the predominant reason why *elpa1* did not perform well for smaller numbers of processors (Fig. 4) because $nb = n/npcol > 64$. If a smaller blocking factor had instead been used, then *elpa1* and *elpa2* would perform similarly and both would outperform *syevd* and *heevd*. Based on this analysis, we opted to have Qbox select *elpa2* if $nb > 64$ and *elpa1* if $nb < 64$ (note that $nb = N_{band}/npcol$).

4.1.1.2 Performance improvement in Qbox simulations using ELPA

The system examined in this comparison was a supercell of silicon carbide $(SiC)_{256}$ with 2,048 electrons, thus matrices of size $2,048 \times 2,048$ are diagonalized. 64 KNL nodes on Theta were used with 64 MPI ranks per node and ranks arranged in a 64×64 configuration (nrowmax=64), thus each column group was contained within a single KNL node. A runtime performance improvement of 4.8x for diagonalization was observed by switching from *syev* to *elpa*, resulting in a 2x improvement of the wave function update kernel. On Cetus with 256 BG/Q nodes, and 64×64 configuration, the changing of *syev* to *elpa* results in an improvement of 7x for diagonalization and an improvement of 2x for the wave function update kernel. Switching from *syev* to *syevd* results in an improvement of 2x and 3x for diagonalization on Theta and Cetus, respectively.

4.1.2 Improving strong scaling by on-the-fly remap data

Fig. 7 shows a high-level runtime decomposition of a hybrid-DFT calculation in Qbox for silicon carbide $(SiC)_{256}$ with 2,048 electrons. The total runtime is mainly spent in the following three kernels:

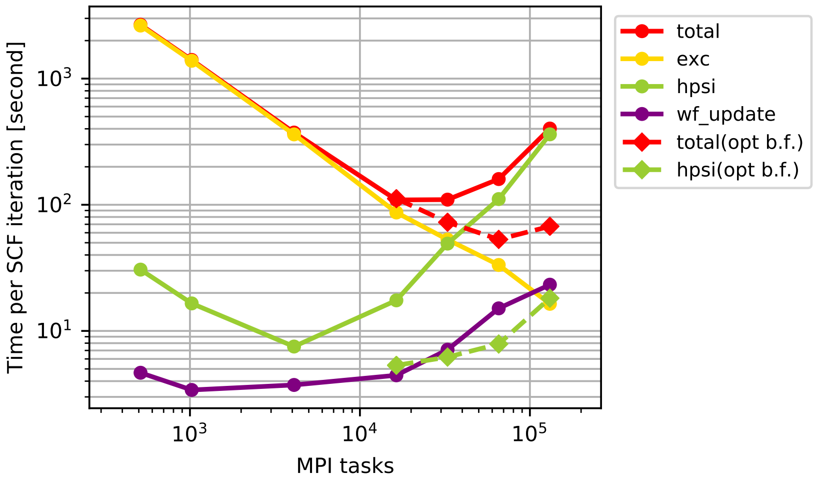


Figure 7: Strong scaling study of Qbox (v 1.63.5) on Theta for the $(SiC)_{256}$ with the PBE0 exchange-correlation functional. Processes are arranged in a 2D $nproc = nprow \times npcol$ grid, where $nprow$ was fixed at 256 and $npcol$ increased. Curves with circles show runtimes from unmodified code, where the total time per iteration (red circles) effectively stops scaling after 16,384 MPI tasks. After optimizing for the blocking factor of matrices, strong scaling performance of Qbox improves considerably (red diamond) with largest speedup coming from the *hpsi* function.

(1) *exc* – exchange-correlation kernel, which includes the computation of the exchange energy,

$$E_x = -\frac{1}{2}\sum_{ij}\int \psi_i^*(\mathbf{r}')\psi_j^*(\mathbf{r}')\frac{1}{|\mathbf{r}-\mathbf{r}'|}\psi_i(\mathbf{r})\psi_j(\mathbf{r}) = -\frac{1}{2}\sum_{\mathbf{G},ij}|P_{ij}(\mathbf{G})|^2 V(\mathbf{G})\,. \tag{7}$$

where ψ_i's are Kohn-Sham orbitals, $P_{ij}(\mathbf{G})$ is the Fourier transform of the pair density $\rho_{ij}(\mathbf{r}) = \psi_i(\mathbf{r})\psi_j(\mathbf{r})$, and $V(\mathbf{G})$ is the Coulomb potential in reciprocal space. Since the wave functions are stored in reciprocal space, one first transforms $\psi(\mathbf{G})$ to $\psi(\mathbf{r})$, computes the pair density in real space, and then transforms the pair density to reciprocal space to compute the exchange energy. The runtime is predominantly spent in 3D FFTs of $\rho_{ij}(\mathbf{r})$ to $P_{ij}(\mathbf{G})$, for which there are N_{band}^2 pairs to evaluate. With the help of a subspace bisection method [25], the number of pairs actually evaluated could be reduced to a fraction of N_{band}^2. However, these 3D FFTs are still a dominant contribution to the runtime. Nevertheless, this kernel scales efficiently up to 131,072 cores as the 3D FFTs are computed independently and distributed across column groups of processors [see the data distribution in Fig. 2].

(2) *hpsi* – applying the Hamiltonian to the wave function, including (a) kinetic energy operator; (b) local potentials (transforming orbitals to real space, multiplying by potential, and transform product back to reciprocal space); (c) nonlocal potentials (including exact exchange correlation functional). In a hybrid-DFT calculation, step (c) is the most expensive part involving matrix-matrix multiplications. Three types of matrix-matrix multiplication operations are involved: $R^T \cdot R'$, $R \cdot S$ and $S \cdot S'$, where R and R' are tall skinny matrices of $n_{pw} \times N_{band}$; S and S' are square matrices of $N_{band} \times N_{band}$, and $N_{band} << n_{pw}$.

(3) *wf_update*: updating wave function coefficients. Several algorithms are currently implemented in Qbox. The algorithm used in Fig. 7 is Preconditioned Steepest Descent (PSD), which

primarily involves *d(z)gemm* and Gram-Schmidt orthogonalization. Other schemes, like JD discussed earlier, involve solving eigenvalue problems (e.g. *syev*, *heev*, or *elpa*).

From Fig. 7, we observed that the *hpsi* and *wf_update* kernels stopped scaling above 16,384 processors (256 KNL nodes). The dramatic increase in runtime for *hpsi* was identified to be from a mismatch of blocking factors used for square and tall-skinny matrices. The blocking factors of tall-skinny matrices (R and R') were set as $mb = n_{pw}/nprow$ and $nb = N_{band}/npcol$ (see Fig. 2). However, the blocking factors for the smaller square matrices (S and S') were set as $mb = nb = 64$. While no noticeable performance difference is observed at smaller processor counts, this mismatch leads to an increased communication cost in *d(z)gemm* at larger processor counts. After adjusting the blocking factors to be $mb = nb = N_{band}/npcol$ for all matrices, the runtime of *hpsi* reduced by about one order of magnitude in the strong scaling limit $nproc > 16,384$ (green diamonds in Fig. 7). Unfortunately, the overall calculation still does not scale ideally above $nproc \sim 16,384$.

The performance bottleneck for ideal strong scaling was observed to come from communication overheads in parallel dense linear algebra kernels (e.g. ScaLAPACK's *d(z)gemm*, *syrk*, and *potrf*). When *nproc* is increased with *nprow* fixed, this increases *npcol*, such that bands are distributed across more column groups and the exact exchange kernel continues to scale. However, this has the undesired effect that the local matrix owned by a processor becomes increasingly more skinny resulting in increased communication costs between column groups. The major issue here is that the processors are logically arranged in a way that is optimal for the exact exchange calculation, but not for matrix operations needed by other kernels. Instead of trying to find a single arrangement of processors that is simultaneously optimal for all kernels, we chose instead to create separate BLACS contexts that optimally map onto the needed matrix operations and redistribute data on-the-fly between the two contexts as needed. We discuss two approaches explored for remapping the data in the following sections: gather & scatter remap and transpose remap.

4.1.2.1 Gather & scatter remap

In the gather & scatter remap, a subgroup of processors are assigned for ScaLAPACK matrix operations while all other calculations (e.g. *exc*) remain assigned to the original large processor grid. In particular, we reduce *nprow*, *npcol*, or both in the original context to create the new context. Data is then exchanged between small and original processor grids using a custom remap function implemented in Qbox. In this case, the runtimes of ScaLAPACK functions in large scale runs are now reduced back to their minimum values as the small grid was chosen based on benchmarking.

Fig. 8 schematically shows the two data layouts when using the gather & scatter remap. In this example, a smaller context of (4×4) processors is created from the original (4×16) context. The data transfer needed between these two contexts only involves communication amongst the four processors in a row. Therefore, we are replacing the costly global communication in the original context with 1) a cheap remap communication within rows plus 2) a cheaper global communication in the smaller context (see Fig. 9).

The ScaLAPACK subroutine *pdgemr2d* was initially investigated for transferring data between the two contexts. Unfortunately, the time spent remapping the data with *pdgemr2d* was similar to the hpsi and wf_update calculations. A detailed profiling study found that the communication pattern in *pdgemr2d* was global (possibly because it is written to support generic use cases) and did not take advantage of the linear mapping between contexts (e.g. re-scaling along single axis). Replacing the global communication pattern with one that takes advantage of the special relation between contexts

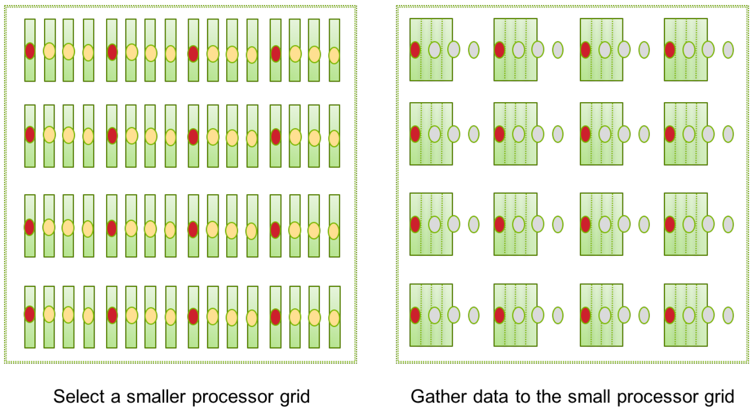

Select a smaller processor grid
Gather data to the small processor grid

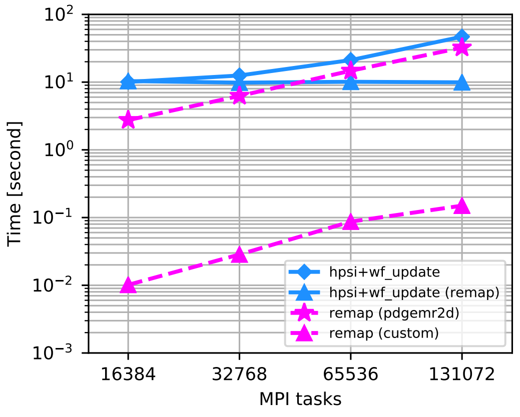
Figure 8: Schematic representation of data layouts before and after application of the gather & scatter remap. The original 4×16 context (left) is remapped into a smaller 4×4 context (right) by remapping data across rows. The circles represent processors with red colored circles depicting those processors that are active in the respective context; gray circles represent idle processors. The green rectangles represent local matrix blocks owned by specific processors. In the original context, each processor owns a single green rectangle, whereas in the smaller context processors own 4 green rectangles from nearby processors in the same row. After the ScaLAPACK computation has completed in the smaller context, the active processors scatter their data back to the idle processors and the original context becomes active again.

was key to reducing the communication cost of the remap stage. As shown in Fig. 9, our custom remap was 1000x more efficient than the generic ScaLAPACK *pdgemr2d* function. When the custom remap is used in a production Qbox calculation, the overall scaling is further improved approaching ideal performance in the strong scaling limit [see triangle curve in Fig. 9(b)]. In this particular case, the runtime per SCF step is reduced to 30.5 seconds compared to 400.8 seconds in original code and 67.5 seconds after optimizing for blocking factors. This significant improvement in strong-scaling performance makes ab initio molecular dynamics simulations using hybrid functionals for extended time scales and/or larger systems more practical on large-scale computing resources.

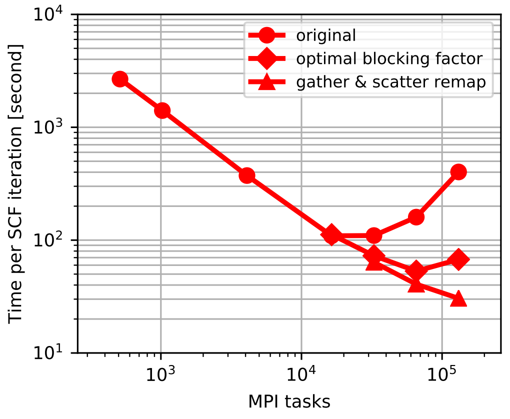
4.1.2.2 Transpose remap

One drawback in the gather & scatter remap is that only a subset of processors are active for the ScaLAPACK subroutines while the rest are idle. In other words, a fraction of computer time is wasted periodically while the calculation runs. In the gather & scatter remap example above, we reduced *npcol* and kept *nprow* unchanged. In fact, since a lot of matrices involved in *d(z)gemm* operations are tall-skinny matrices, increasing *nprow* might help further to reduce ScaLAPACK's runtime. This is indeed the case for $(\mathrm{SiC})_{256}$ as demonstrated in Fig. 10(a). It is clear that for a fixed *npcol*, increasing *nprow* helps to reduce the ScaLAPACK runtime. Therefore, we alternatively could have created a context with the same number of active processors as the original context but with both a larger *nprow* and smaller *npcol*. This is the essential idea of transpose remap discussed next.

The transpose remap is schematically shown in Fig. 11. In this example, we map the original 4×4 grid into an 8×2 grid. The processors are rearranged in a way such that data movement between the two contexts is minimal. In particular, the 2D coordinates of a processor in the new context $(myrow', mycol')$ are related to coordinates in the original context $(myrow, mycol)$ in the following



(a)



(b)

Figure 9: Improvement of strong scaling in Qbox using the gather & scatter remap for $(SiC)_{256}$ with the PBE0 exchange-correlation functional. (a) Remap times (*pdgemr2d* and custom) compared with *hpsi* and *wf_update* before and after optimization. (b) Summary of strong scaling performance improvements of Qbox on Theta comparing original code (circles), optimization of blocking factors (diamonds), and remap (triangles).

way,

$$myrow' = myrow \times g + mod(mycol, g), \quad mycol' = mycol/g\,, \tag{8}$$

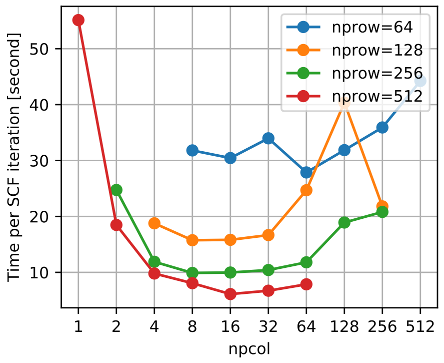
where g is a shrinking factor in the column dimension. The blocking factor is correspondingly changed as follows,

$$mb' = mb/g, \quad nb' = nb \times g\,. \tag{9}$$

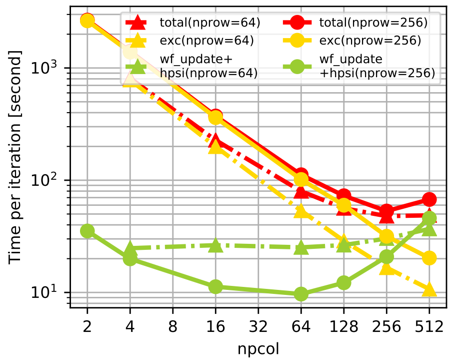
The communication pattern is again local in the transpose remap with each process only communicating with g nearby processes within the same row (in the original context). In fact, this data movement is essentially a series of transposes within non-overlapping subgroups of g processes. We again expect that a custom implementation of the transpose remap communication will perform better than the generic ScaLAPACK *pdgemr2d*.

However, the transpose remap involves an additional constraint. The data distribution in Fig. 11 can only be obtained if $mod(mb, g) = 0$ in Eq. (9). If this condition is not met, there will be data movement between processes of different rows resulting in a more complicated communication pattern. To circumvent this, in the initialization of the dimension of the wave function, we adjust m such that $mod(m, nprow) = 0$ and $mod(m/nprow, g) = 0$. This is done in the code by transforming the wave function to real space, adjusting the real space grid and interpolating to obtain values on the new grid, and then transforming back to the reciprocal space.

The transpose remap will especially benefit those cases where the exact exchange kernel prefers small $nprow$. In the case of $(SiC)_{256}$, the FFT grid size $256 \times 256 \times 256$, which could scale efficiently up to 256 processes. However, due to the ratio of KNL computational/memory performance relative to network bandwidth of the Aries interconnect on Theta, we observed that these relatively small 3D FFTs did not scale well beyond multiple KNL nodes. This is seen in Fig. 10 where *exc* is faster



(a)



(b)

Figure 10: Runtime performance of different contexts in Qbox on Theta for the $(SiC)_{256}$ system. (a) total runtime of ScaLAPACK subroutines involved in single SCF iteration with different nprow and npcol. (b) runtime of Qbox kernels for different *nprow* (nrowmax).

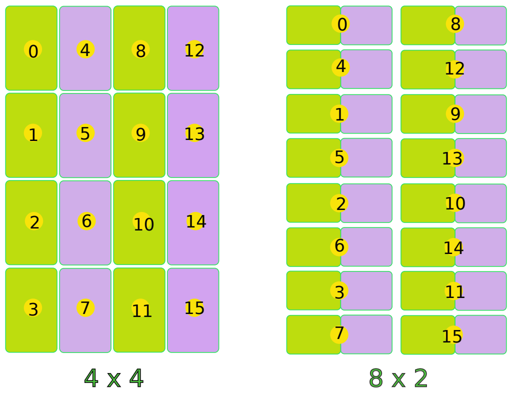


Figure 11: Schematic demonstration of data layouts before and after application of the transpose remap. The original 4×4 context (left) is remapped into a new 8×2 context (right) with all processors active. The yellow circles represent the processors and numbers represent the processor id. The rectangle blocks represent the local matrices owned by the processors.

if $nprow = 64$ compared to 256 for identical values of $npcol$ (i.e. yellow triangles always faster than yellow circles). However, the problem is that if $nprow = 64$ is used, then the ScaLAPACK runtime becomes larger than the case of $nprow = 256$ [see Fig. 10(a)]. If we were to use the gather & scatter remap, then there would not be too much of an improvement since $nprow' \leq nprow$. However, using the transpose remap, we are able to increase $nprow$ reducing ScaLAPACK's runtime.

Finally, Table 1 shows the further improvement achieved by using transpose remap with $nprow = 64$. As can be seen, using the transpose remap results in the lowest observed runtime of 19.2 seconds per iteration on 512 KNL nodes on Theta. Using the MKL library for 3D FFTs, the runtime is further reduced to 15.0 seconds per SCF iteration. For the case of $nprow = 256$, since 3D FFTs operations are dominated by communication time, changing math libraries does not improve their performance.

Table 1: Performance improvement obtained by choosing consistent blocking factors (opt. b.f.) and using remap methods ($npcol' = npcol/8$): the time shown is the total runtime per SCF iteration in seconds. Cray LIBSCI library was used in this benchmark.

$nprow$ (N_{nodes})	original	opt. b.f.	gather & scatter	transpose
256 (2048)	400.81	67.45	30.52	29.93
64 (512)	369.81	47.92	36.62	19.21

4.1.2.3 Suggestions on how to set *nrowmax* and *remap* parameters

Some guidance to setup contexts for large scale hybrid-DFT calculations in Qbox follows:

(1) determine $nrowmax$: this controls $nprow$ in the original process grid. One could choose $nrowmax$ to be the maximum number of processes for optimal performance of 3D FFTs. On BG/Q, a good choice is to set $nrowmax = N_z$ where N_z is 3D FFT grid size along z direction. On Theta, since relatively small 3D FFTs do not scale well beyond a single KNL node (compute/memory vs. communication), a good choice is to set $nrowmax = 64$ (number of cores per KNL node assuming 1 MPI rank per core).

(2) determine $remap$: once $nrowmax$ and $nprow$ are known, $npcol = nproc/nprow$. The blocking factor in the column dimension is determined as $nb = N_{band}/npcol$. If $nb > 16$, it is generally not necessary to remap data, however, if $nb < 16$, then one could set $npcol' = npcol \times nb/16$, such that in the new context $nb \geq 16$. One could then decide whether to use the gather & scatter or transpose remap strategies based on $mb' = m/nprow'$ ($nprow' = nproc/npcol'$). If the new blocking factor $m/nprow'$ is larger than 16, transpose remap may be beneficial, otherwise one should just use gather & scatter remap.

By using these remap strategies, the strong scaling limit of the whole hybrid-DFT calculation is $nproc = N_z \times N_{band}/2$ (or $nproc = 64N_{band}/2$ as suggested for Theta). The factor 1/2 is due to the fact that the subspace bisection algorithm will not reduce the number of nonzero overlap pairs in the limit of one band per column group. For LDA or GGA calculations, it is challenging to scale the overall calculation to $nproc = N_z \times N_{band}/2$ (or $nproc = 64N_{band}/2$ on Theta). Therefore, we would not suggest to use remap in LDA and GGA. Instead, one could directly set the original context to be optimal for ScaLAPACK computation as 3D FFTs are generally a small part of the total runtime.

4.2 Optimization of WEST

This section summarizes all the optimization accomplished with the WEST codes during our Theta ESP project. We start with single node benchmarks on different platforms, where the majority of runtime is observed to be spent in LAPACK and 3D FFT libraries. Similar to Qbox, WEST primarily relies on the highly performant external libraries to fully utilize the advanced features of an architecture. Given this, we again focus on the parallelization of the code, in particular, using band and task group parallelizations to extend the strong-scaling limit of WEST. I/O operations were improved by adopting a single-reader-broadcast scheme and by implementing support for base64 encoding, which facilitates portability across different endian machines.

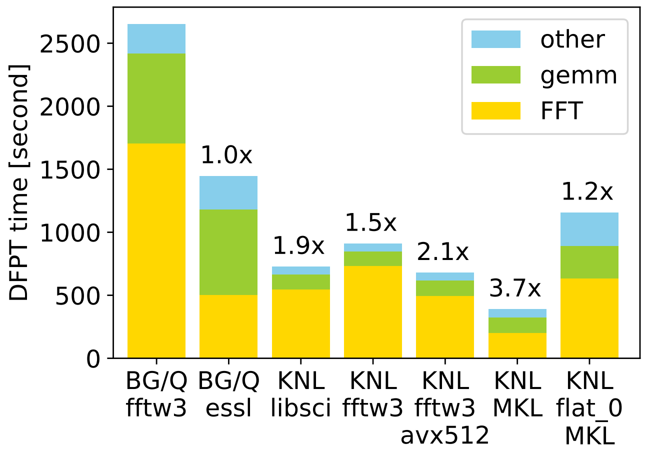


Figure 12: Performance comparison between BG/Q and KNL linking *wstat* to different libraries (FFTW3 and ESSL on BG/Q; LIBSCI, MKL, and FFTW3 on KNL). The system examined is a CdSe nanoparticle with 884 electrons. The labels *fftw3-avx512* and *fftw3* correspond to the fftw-3.3.6 library compiled with and without, respectively, AVX-512 for FFTs. On Theta system, the label *KNL-flat_0-MKL* corresponds to running in flat memory mode (using DDR); otherwise, the cache memory mode was used.

4.2.1 Performance of external libraries: single perturbation study

We initially focused on single-node performance of *wstat*. Fig. 12 shows the runtime for the calculation of the response of one perturbation using different platforms (BG/Q and KNL) and FFT libraries (ESSL, FFTW3, and MKL). The computation on Theta is performed on a single KNL node with 64 cores, whereas the computation on BG/Q is performed on 4 BG/Q nodes with a total of 64 cores. As shown in Fig. 12, the runtime of *wstat* is predominantly 3D FFTs and linear algebra operations (GEMMs).

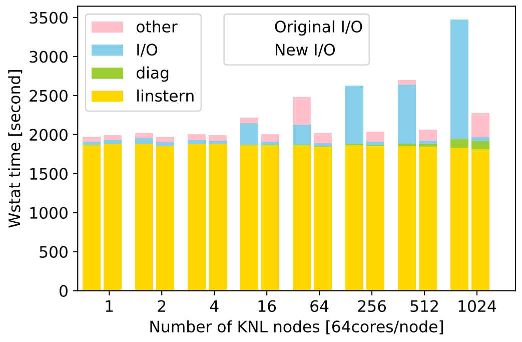
(a) FFT: 3D FFTs of wave functions and density. For BG/Q, ESSL is about 3x faster than FFTW3. On KNL, MKL results in the best runtime, which is about 2.7x faster than LIBSCI. We also observe that the performance of FFTW3 compiled with AVX-512 is comparable to LIBSCI, and is slightly faster than FFTW3 without AVX-512.

(b) gemm: LAPACK subroutines for multiplication of dense matrices, namely D(Z)GEMM. The performance on KNL depends on the choice of memory mode with allocation into MCDRAM performing 2.1x faster than allocation into DDR memory (KNL-flat_0).

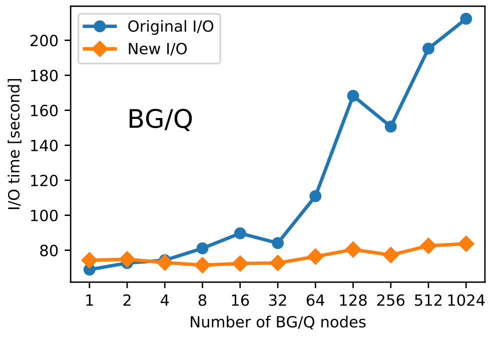
(c) other: MPI All_reduce of distributed arrays (e.g. computing change of density) and remainder of application.

Comparing the cases with best vendor-provided libraries (e.g. ESSL on BG/Q, and MKL on KNL), a factor of 3.7x speedup is observed going from BG/Q to KNL. On both platforms 3D FFTs and D(Z)GEMMs take more than 80% of the total runtime. As one would expect the vendor supplied math libraries already effectively utilize the hardware features of KNL. As we have mentioned, MCDRAM plays important role in the performance as is seen from the comparison between cache memory mode and flat memory mode (using DDR). The original runtime in cache mode was observed to be 3x faster than that in flat mode (DDR). MCDRAM especially benefits performance of memory-bandwidth limited kernels, such as 3D FFT. We note that the KNL in the cache memory mode can access 192 GB DDR memory while retaining the 16 GB MCDRAM as a last-level cache. This KNL feature enables the calculation of the smallest independent unit of work (single perturbation in this case) of a large-scale WEST calculation to be tractable within a single KNL node. This is in contrast to the BG/Q system where the memory is limited to 16 GB per node and one may be forced to spread the calculation across multiple nodes.

4.2.2 I/O optimization



(a)



(b)

Figure 13: Time spent reading the initial wave function initially observed to increase with number of KNL nodes on Theta with Lustre filesystem (a) becoming a significant fraction of runtime. The same effect was observed to a lesser extent on BG/Q with GPFS filesystem (b). Time spent in I/O reduced to negligible fraction of runtime on 1-1024 nodes by having master process read and distribute wave function. The number of perturbations is twice of the number of nodes.

As mentioned in Sec. 3.2, the implementation of the Davidson diagonalization algorithm is embarrassingly parallel in the perturbation index, however the near-ideal scaling can be compromised by I/O operations. In order to assess the scaling of I/O operations in *wstat*, we performed a weak scaling benchmark study on Theta by computing the response of the system to a constant number of perturbations per KNL node, therefore by increasing the number of perturbations and processors at the same time ($N_{pert} = 2\,N_{node}$). The benchmark results on Theta (featuring a Lustre filesystem) showed that I/O operations are responsible for the weak-scaling degradation [see Fig. 13(a)] and, to a lesser extent, a similar performance degradation is observed on Mira (featuring a GPFS filesystem) [see Fig. 13(b)].

After carefully profiling the code (see CPU trace profiles in Fig. 14), we determined that the I/O performance degradation can be prevented by restructuring the initial I/O of *wstat*. As a matter of fact, in order to compute the response to a given perturbation, *wstat* needs to load the DFT electronic structure (energies and wavefunctions), however such information does not depend on the perturbation and can therefore be read from the filesystem by single replica and then broadcast to all other replicas.

With the new single-reader-broadcast I/O scheme, the scaling of the initial I/O phase substantially improved resulting in a cost that remains negligible on both Theta [Fig. 14(d)] and BG/Q [Fig. 14(b)] up to 1024 nodes.

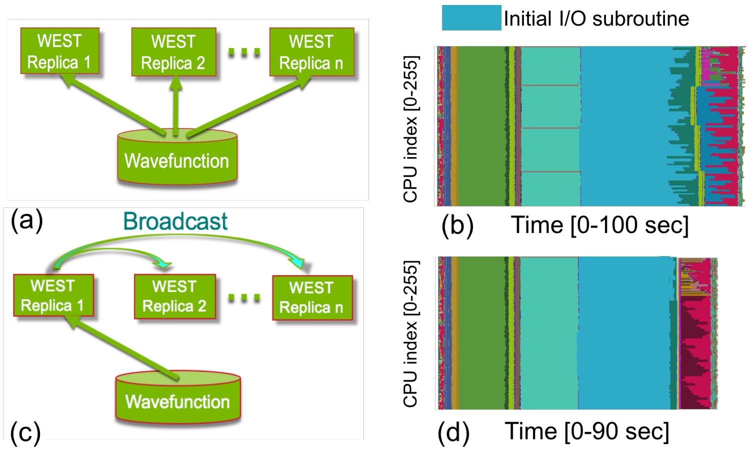


Figure 14: Restructuring the I/O distribution scheme in *wstat*: (a) original I/O scheme where all replicas read checkpoint files simultaneously; (b) CPU trace for the original I/O scheme where many replicas (4 in this case) observed to interfere with each other degrading performance; (c) new I/O scheme where a single replica reads checkpoint file and broadcasts data to all other replicas; (d) CPU trace for new I/O scheme.

The profiling of I/O operations on Theta has also revealed a performance bottleneck of the library used by WEST to write XML files, the IOTK library (distributed within Quantum Espresso 5.4.0). Fig. 15 shows the read and write performance of IOTK for a single complex array of varying size with the IBM XL compiler on BG/Q and two separate KNL installations at Argonne using the Intel compiler (Theta at ALCF and nodes at JLSE). Performance of the IOTK library is compared with direct READ and WRITE functions provided by FORTRAN. As shown in the figures, direct READ and WRITE is faster than IOTK in all cases with the largest performance difference observed on Theta (Lustre) where direct READ is about three orders of magnitude faster than IOTK.

The following line in the IOTK library was found responsible for the I/O performance bottleneck:

```
READ(unit, iostat=iostat) (dat(j), j = 1, ubound(dat, 1))
```

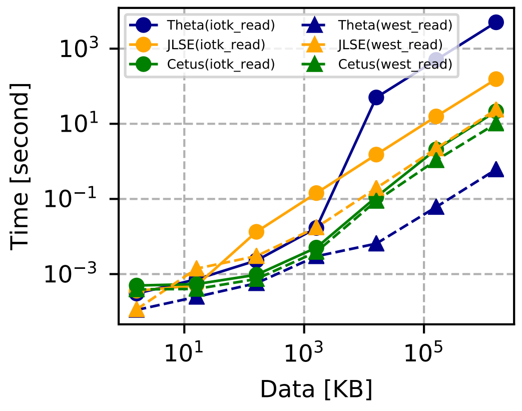
which loops over elements of the array. In the direct READ/WRITE implementation, the previous statement was simply replaced with the following line (to assist the compilers).

```
READ(unit, iostat=iostat) dat
```

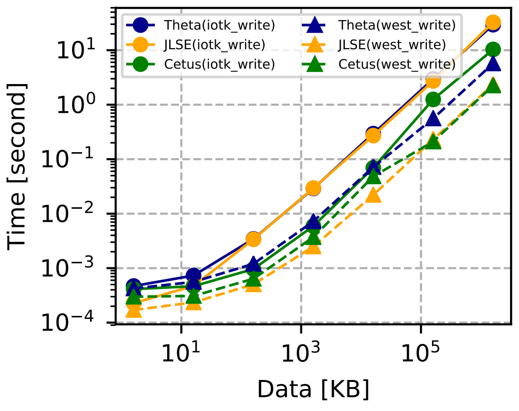
In all cases, this relatively straightforward change improves the I/O performance on all three systems (Cetus, JLSE, and Theta) for reading and writing regular arrays of data in checkpoint files. These improvements with direct READ and WRITE also significantly improved I/O performance in other portions of WEST, including I/O that frequently occurs during calculations in *wfreq*.

4.2.3 Parallelization

In order to exploit the embarrassing parallelism of Eq. (5), at the basis of the linear response calculation, the WEST 2.0.0 implementation features two layers of parallelization: over (1) plane



(a) Read



(b) Write

Figure 15: Benchmark of I/O performance for different size of data on Cetus (BG/Q), Theta (KNL), and JLSE (KNL). The subroutines iotk_read() and iotk_write() read and write, respectively, using the default iotk library, whereas the subroutines west_read() and west_write() use direct READ and WRITE functions provided by FORTRAN.

waves and (2) external perturbations. These algorithm decisions set the strong-scaling limit of *wstat* to $N_{pert} \times N_z$ processes, where N_z is the number of 2D FFT slices along the z-axis and N_{pert} is the number of perturbations. This scaling limit is demonstrated in Fig. 16 for a silicon nanocrystal ($Si_{35}H_{36}$) calculation, in which performance scales efficiently only up to 256 KNL nodes (64 cores per node) for 256 perturbations with one perburbation per node [see yellow circle curve on Fig. 16].

We have added a third layer of parallelization with the purpose of distributing the sum over single particle states necessary to compute the total charge density. We note indeed that Eq. 6 involves the calculation of independent single-particle responses. With this additional layer of parallelism, the calculation of the single-particle response is assigned to N_{band} band-groups, and the total variation of the density $\Delta\rho_i$ is obtained with a MPI_Allreduce operation. The latter operation has a negligible cost with respect to the calculation of single particle responses, and overall band parallelism results in a significant improvement of strong scaling performance (see Fig. 16). Band parallelization extends the strong-scaling limit of *wstat* algorithm up to $N_{pert} \times N_{band} \times N_z$ cores. This improvement enables *wstat* to readily utilize the full Theta machine and paves the way to full Aurora (next generation ALCF leadership supercomputer) calculations for systems with 2–5 thousand electrons.

One limitation of the current band parallelization implementation is that every band group owns an entire copy of all data including wave functions and non-local pseudopotentials. In this case, as the system size increases to unprecedented sizes for GW calculations, this data will eventually no longer fit within the memory of a single KNL node (192 GB per node on Theta) and it will be necessary to spread individual band groups across multiple KNL nodes, incurring performance degradation caused by FFT inter-node communication. To address these cases, we have explored the possibility to distribute independent FFT operations on a smaller number of cores. This protocol is termed task-group parallelization and is used in Quantum Espresso to extend the scalability of 3D FFT beyond N_z processes. In WEST task-group parallelization can be used to redistribute different 3D FFT operations over a smaller number of processors so that to avoid expensive inter-node

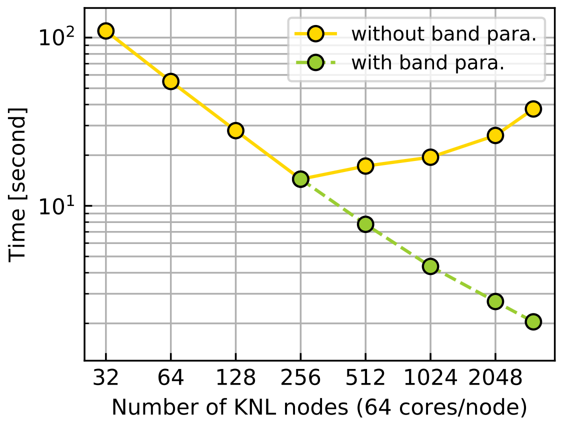


Figure 16: Strong-scaling performance of *wstat* without (yellow) and with (green) band parallelization (BP) on Theta. The system studied is a silicon nanoparticle $Si_{35}H_{36}$ with 176 electrons and 256 perturbations.

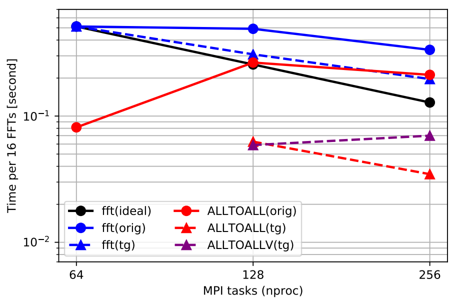
communication operations. In other words, the whole band group is partitioned into smaller task groups, each of which fits within a single node, thus enabling 3D FFTs to be computed within a single KNL node taking advantage of shared memory MPI communication for transposes. However, similar to the remap strategies implemented in Qbox, this is beneficial only if the communication overhead due to the data redistribution step is faster than the original MPI_Alltoall.

Fig. 17 shows the performance of 3D FFT operations performed on 16 wave functions with FFT grid sizes of 256^3 and 512^3. Without using task groups, the MPI_Alltoall (solid red curves) communication operations involved by 3D FFT was identified as the largest fraction of runtime. After redistributing wave functions using task groups, the MPI_Alltoall time is significantly reduced in multiple-node cases; however, the overhead due to redistributing data via MPI_Alltoallv (purple curves in Fig. 17(a) and (b)) is slightly cheaper than original MPI_Alltoall call. This is primarily due to two reasons: (1) the amount of data transferred per wave function in MPI_Alltoallv is smaller than the original MPI_Alltoall and (2) data for several wave functions (equal to the number of task groups) are distributed together, whereas in the original case each MPI_Alltoall is completed for single wave functions leading to increased latency effects.

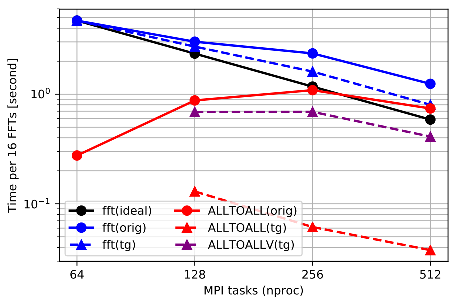
By differently striding MPI ranks in task groups, we have tested whether it is beneficial to pack within the same node all processes that belong to the same task-group. This rearrangement does not have a significant effect on the overall runtime, although individual stages are affected [see Fig. 17 (c) and (d)].

Table 2 shows runtimes for full *wstat* calculations with a relatively large nanoparticle consisting of 301 atoms and 2816 electrons. The 3D FFT grid for this calculation is $480 \times 480 \times 480$ and 8 KNL nodes were used with one MPI rank per core. With the use of task groups, the total 3D FFT runtime is improved by $\sim$40%, resulting in an overall improvement of the total runtime by 20%.

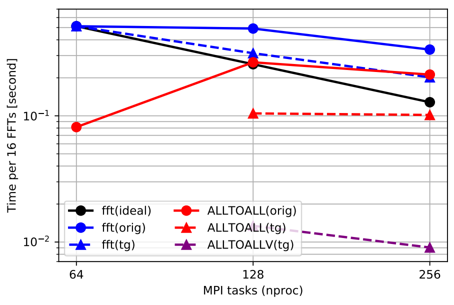
Finally, we have identified in *wfreq* a region of the code which could be better parallelized. The calculation of the macroscopic dielectric constant involves the calculation of $3\,N_{occ}$ independent Sternheimer equations. Using ideas similar to those discussed in earlier sections, the independent



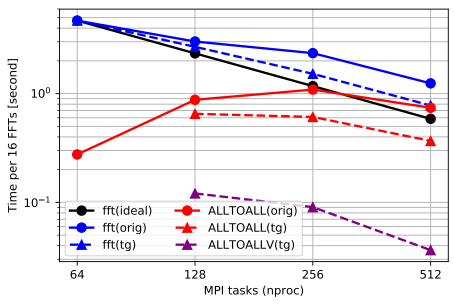
(a) $N = 256$, rearranged processors



(b) $N = 512$, rearranged processors



(c) $N = 256$, no rearrangment



(d) $N = 512$, no rearrangment

Figure 17: Performance of task group parallelism in *wstat*. The reported runtime is for the forward and backward 3D FFT transformation of 16 wave functions with FFT grid sizes of 256^3 and 512^3. The solid curves are without task groups and dashed curves are with ntask_group = num_node = nproc/64. In (a) and (b), the processors in band groups are arranged such that all the processors in a task group will be within the same node; whereas in (c) and (d) the original processor arrangement is retained.

equations were distributed across the parallelism layer that also distributes the perturbations. This leads to a speed-up for this part of the calculation proportional to the size of the MPI group that distributes perturbations, which theoretically is $\leq N_{pert}$. As the solution of these Sternheimer equations is needed by all perturbations in order to proceed further, an MPI_Allreduce operation was introduced, adding a negligible cost to the overall computation.

5 Portability

In general, the initial porting effort of Qbox and WEST to the Intel KNL architecture on Theta was straightforward as both codes routinely run on Intel Xeons and typically large fractions of runtime are spent in external math libraries to compute 3D FFTs and dense linear algebra (e.g. FFTW, ESSL, MKL, ScaLAPACK, and ELPA). Initially changing from BG/Q to KNL primarily involved setting appropriate compiler flags and linking appropriate vendor-provided math libraries. The AVX-512 vectorization enabled by KNL was principally taken advantage of through external math libraries and the Intel compiler. For problem sizes with sufficiently small memory footprints (< 16 GB per node), the MCDRAM was utilized in the flat-quadrant memory-cluster mode after setting

Table 2: Performance comparison using task groups in *wstat* with 8 KNL nodes on Theta.

kernel	without task group	8 task groups (rearranged)	8 task groups (not rearranged)
Total WSTAT	1h58m	1h35m	1h41m
3D FFT	3729.23s	2294.99s	2378.65s
MPI_AlltoAll	2500.48s	93.25s	1154.16s
Data movement	–	996.60s	45.34s

"numa -m 1". For larger problem sizes, the MCDRAM was utilized via the cache-quadrant memory-cluster mode. Additional libraries, such as xerces-c and ELPA, were straightforward to build on Theta given the similarity of the Cray programming environment on Theta to other computational resources.

A major focus of our optimization effort for Theta was that of reducing communication overheads in both Qbox and WEST, in part due to the increased computational intensity and memory bandwidth relative to network bandwidth compared to the relatively balanced IBM BG/Q (e.g. Mira at ALCF). All of the optimized algorithms implemented during this project are portable to other architectures as evidenced by improved performance also observed in large-scale calculations on Mira. However, due to differences in technical specifications across different computational resources (e.g. interconnect), we expect some minimal benchmarking to be necessary to setup optimal configurations for both codes to efficiently use those resources. For example, on the BG/Q systems, 3D FFT calculations in Qbox and WEST scale ideally to number of cores equal to the grid size along the z-axis (i.e. one slab per core). On Theta, however, degraded scaling was observed for similar 3D FFT calculations across multiple KNL nodes due to the increased ratio of memory to network bandwidth (albeit with reduced runtimes compared to BG/Q). With benchmarking and appropriately selecting configuration settings (e.g., *nprow* in Qbox and size of band/task groups in WEST), improved performance can be obtained.

Additionally, to enable workflows to utilize both Mira and Theta at ALCF, Base64 encoding/decoding was implemented for checkpoint files in WEST. This makes the input, output, and intermediate binary files transferable between different endian systems (e.g. Mira and Theta), making it possible to utilize both resources, as needed, to run large-scale science campaigns.

6 Conclusions

In summary, we have optimized two open-source electronic structure codes, WEST and Qbox, on Theta. As a result of this Theta ESP project, the weak- and strong-scaling performance was significantly improved for the two codes on Theta where the ratio of compute/memory capabilities relative to network are larger than recent architectures. In addition, many of the optimizations implemented here have also improved performance on relatively balanced architectures (e.g. IBM BG/Q), where both codes already had good performance. Thus, utilization of large fractions of available leadership computing resources combined with improved time-to-solutions enabled by this project are enabling science-critical calculations that were not previously possible.

For Qbox, we implemented data remap methods to efficiently redistribute data on-the-fly between ScaLAPACK contexts that are chosen to be optimal for respective operations on tall-skinny and small-symmetric matrices. These algorithms enable efficient calculation for separate phases of Qbox, where it is challenging to use a single context to solve rather different computational tasks. The performance improvements from these optimizations have enabled large-scale exact exchange calculations with good parallel efficiency in the limit of one band per column group. This capability significantly reduces the time-to-solution for hybrid-DFT calculations and enabled efficient utilization of 100% of Theta, for example, thus enabling ab initio molecular dynamics simulations for systems with thousands of electrons.

For WEST, the initial I/O phase was restructured to use a single-reader-broadcast algorithm, which significantly improved weak scaling performance on the full Theta resource. Additional levels of parallelism were implemented (band- and task-based parallelization) that exposed finer-grain work units that extended the strong scaling limits of the implemented algorithms. These optimizations widen the scope of possible excited state calculations that are able to efficiently utilize large fractions of today's leadership computing resources today (e.g. 100% Theta) and facilitates design of scientific campaigns to potentially utilize large fractions of upcoming large-scale resources (e.g. an exascale resource).

As the majority of the optimization effort in this project focused on addressing communication overheads, it is fully expected that these portable improvements will also help to improve performance of smaller-scale calculations. Additionally, with the current trend of computational intensity increasing faster than the rate at which data can be moved between nodes, addressing the communication overheads now will place both codes in a better position to utilize larger-scale resources of the near future. Some of the guiding principles of our portable optimization efforts here to reduce internode communication costs and improve overall scalability of both applications are expected to be generally applicable: (1) appropriately distribute independent work units (independent in the sense of minimal inter-work communication); (2) reducing or hiding communication overheads by optimizing communication patterns, possibly through rearranging data layouts on-the-fly for different phases of a calculation.

7 Acknowledgment

This work was supported by MICCoM, as part of the Computational Materials Sciences Program funded by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, Materials Sciences and Engineering Division. This research used resources of the Argonne Leadership Computing Facility, which is a Department of Energy Office of Science User Facility under Contract DE-AC02-06CH11357. We thank Vitali Morozov, Sudheer Chunduri, and Michael D'Mello (Intel) for many helpful discussions relating to MKL and 3D FFTs.

References

[1] Richard M. Martin. *Electronic Structure: Basic Theory and Practical Methods.* Cambridge University Press, 1 edition, October 2008.

[2] Vladimir I. Anisimov, F. Aryasetiawan, and A. I. Lichtenstein. First-principles calculations of the electronic structure and spectra of strongly correlated systems: the LDA + U method. *Journal of Physics: Condensed Matter*, 9(4):767, 1997.

[3] F. Aryasetiawan and O. Gunnarsson. The GW method. *Reports on Progress in Physics*, 61(3):237, 1998.

[4] W. M. C. Foulkes, L. Mitas, R. J. Needs, and G. Rajagopal. Quantum monte carlo simulations of solids. *Rev. Mod. Phys.*, 73:33–83, Jan 2001.

[5] Attila Szabo and Neil S. Ostlund. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*. Courier Corporation, June 2012.

[6] W. Kohn and L. J. Sham. Self-Consistent Equations Including Exchange and Correlation Effects. *Physical Review*, 140(4A):A1133–A1138, November 1965.

[7] David C. Langreth and M. J. Mehl. Beyond the local-density approximation in calculations of ground-state electronic properties. *Physical Review B*, 28(4):1809–1834, August 1983.

[8] A. D. Becke. Density-functional exchange-energy approximation with correct asymptotic behavior. *Phys. Rev. A*, 38:3098–3100, Sep 1988.

[9] John P. Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized Gradient Approximation Made Simple. *Physical Review Letters*, 77(18):3865–3868, October 1996.

[10] Jianmin Tao, John P. Perdew, Viktor N. Staroverov, and Gustavo E. Scuseria. Climbing the density functional ladder: Nonempirical meta˘generalized gradient approximation designed for molecules and solids. *Phys. Rev. Lett.*, 91:146401, Sep 2003.

[11] Carlo Adamo and Vincenzo Barone. Toward reliable density functional methods without adjustable parameters: The PBE0 model. *The Journal of Chemical Physics*, 110(13):6158–6170, April 1999.

[12] Jochen Heyd, Gustavo E. Scuseria, and Matthias Ernzerhof. Hybrid functionals based on a screened Coulomb potential. *The Journal of Chemical Physics*, 118(18):8207–8215, May 2003.

[13] Francois Gygi, Erik W. Draeger, Martin Schulz, Bronis R. De Supinski, John A. Gunnels, Vernon Austel, James C. Sexton, Franz Franchetti, Stefan Kral, Christoph W. Ueberhuber, and others. Large-scale electronic structure calculations of high-Z metals on the BlueGene/L platform. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 45. ACM, 2006.

[14] Francois Gygi. Large-scale first-principles molecular dynamics: moving from terascale to petascale computing. *Journal of Physics: Conference Series*, 46(1):268, 2006.

[15] Francois Gygi. Architecture of Qbox: A scalable first-principles molecular dynamics code. *IBM Journal of Research and Development*, 52(1):137–144, 2008.

[16] Mark S. Hybertsen and Steven G. Louie. Electron correlation in semiconductors and insulators: Band gaps and quasiparticle energies. *Physical Review B*, 34(8):5390, 1986.

[17] Lars Hedin. New Method for Calculating the One-Particle Green's Function with Application to the Electron-Gas Problem. *Physical Review*, 139(3A):A796–A823, August 1965.

[18] Marco Govoni and Giulia Galli. Large Scale GW Calculations. *Journal of Chemical Theory and Computation*, 11(6):2680–2696, June 2015.

[19] Hugh F. Wilson, Fran çois Gygi, and Giulia Galli. Efficient iterative method for calculations of dielectric matrices. *Phys. Rev. B*, 78:113303, Sep 2008.

[20] Hugh F. Wilson, Deyu Lu, Fran çois Gygi, and Giulia Galli. Iterative calculations of dielectric eigenvalue spectra. *Phys. Rev. B*, 79:245106, Jun 2009.

[21] Stefano Baroni, Paolo Giannozzi, and Andrea Testa. Green's-function approach to linear response in solids. *Physical Review Letters*, 58(18):1861–1864, 1987.

[22] Stefano Baroni, Stefano de Gironcoli, Andrea Dal Corso, and Paolo Giannozzi. Phonons and related crystal properties from density-functional perturbation theory. *Reviews of Modern Physics*, 73(2):515–562, 2001.

[23] Andreas Marek, Volker Blum, Rainer Johanni, Ville Havu, Bruno Lang, Thomas Auckenthaler, Alexander Heinecke, Hans-Joachim Bungartz, and Hermann Lederer. The elpa library - scalable parallel eigenvalue solutions for electronic structure theory and computational science. *J. Phys. Condens. Matter*, 26:1–15, May 2014.

[24] A. Marek, V. Blum, R. Johanni, V. Havu, B. Lang, T. Auckenthaler, A. Heinecke, H.-J. Bungartz, and H. Lederer. The ELPA library: scalable parallel eigenvalue solutions for electronic structure theory and computational science. *Journal of Physics: Condensed Matter*, 26(21):213201, 2014.

[25] Francois Gygi and Ivan Duchemin. Efficient computation of Hartree-Fock exchange using recursive subspace bisection. *Journal of Chemical Theory and Computation*, 9(1):582–587, 2013.